Linear and Logistic Regression

Priit Järv

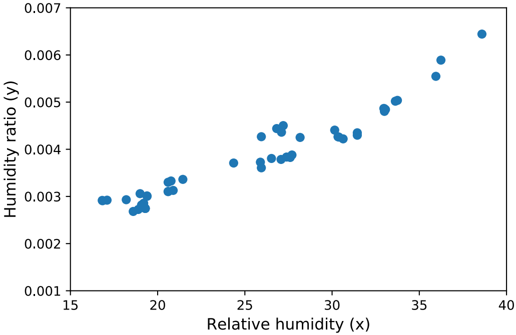
Tallinn University of Technology

priit.jarv1@ttu.ee

May 2, 2019

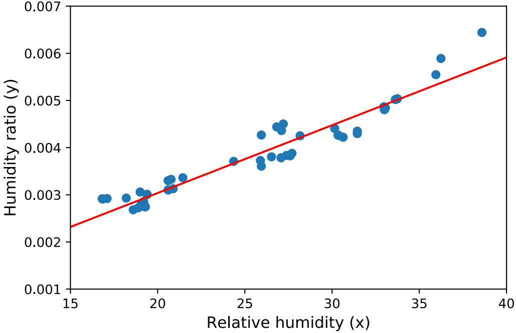
Fitting Linear Functions

Start with some highly correlated data:



Given x, should be easy to predict y.

Fitting Linear Functions



Pretty good estimate ($\hat{y} \approx y$): $\hat{y} = w_0 + w_1 x$

Here, $w_0 = 0.00016$ and $w_1 = 0.00014$

Linear Regression

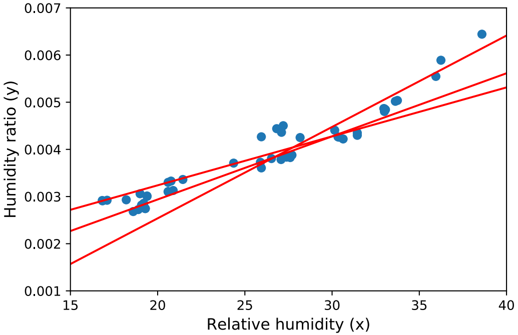
Estimating $\hat{y} = w_0 + w_1 x_1 + \ldots + w_n x_n$ is called **linear regression**

Terminology (depending who you talk to):

$x_1, x_2, \ldots$	y
independent variables	dependent variable
predictors	outcome
features	class[1]
attributes	label[1]

[1] in binary classification

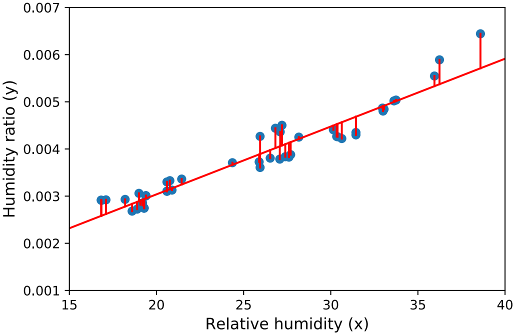
Linear Regression



Many "regression lines" are possible

Which one do we choose?

Linear Regression



Try to minimize this error ϵ, distance to data points

(True function is $y = w_0 + w_1 x + \epsilon$)

Loss Function

We have a dataset of N points
Error for one data point (j) is $y_j - \hat{y}_j$

Loss function represents, how good is our approximation over many data points

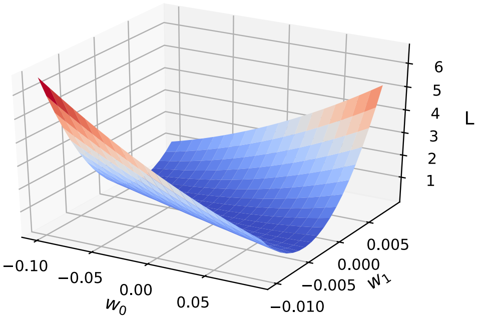
$$L = \sum_j^N (y_j - \hat{y}_j)^2$$

Minimize this sum of squared errors (SSE)

Why squared? Gives the most likely values for w_0 and w_1, if x has normal distribution.

Loss Function

Loss changes as we change the parameters w_0, w_1:



Loss function is convex $\implies$ find a minimum, it will be the global minimum

Least Squares

Minimum of a convex function $y = f(x)$:
find x such that $y' = 0$
Applies to $L = f(w_0, w_1)$, just look at variables separately

Solve:

$$\begin{cases} \frac{\partial}{\partial w_0} \sum_j^N (y_j - (w_0 + w_1 x_j))^2 = 0 \\ \frac{\partial}{\partial w_1} \sum_j^N (y_j - (w_0 + w_1 x_j))^2 = 0 \end{cases}$$

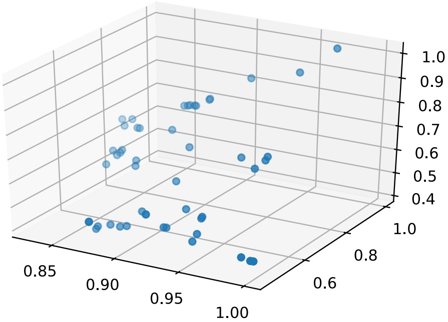
This is called the **least squares** method.

Solution:

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \quad w_0 = \frac{1}{N}\Big(\sum y_j - w_1 \sum x_j\Big)$$

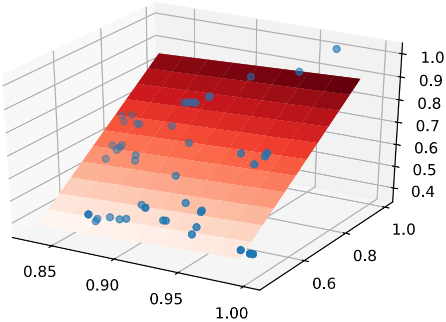
Multivariate Regression

What if y depends on multiple variables?



Input data (vector): $\mathbf{x} = x_1, x_2, \ldots, x_n$
Weights (vector): $\mathbf{w} = w_0, w_1, w_2, \ldots, w_n$

Multivariate Regression



Fit a hyperplane:
$\hat{y} = w_0 + w_1 x_1 + \ldots + w_n x_n = w_0 + \sum_j^n w_j x_j = \mathbf{w} \cdot \mathbf{x}$
(Here, $n = 2$. Use dummy $x_0 = 1$ for convenience)

Gradient Descent

Solving for **w** such that $L' = 0$ is expensive with multiple variables
Time to approximate (as usual).

Start with some vector **w**, compute $\frac{\partial L}{\partial w_0}, \frac{\partial L}{\partial w_1}, \ldots$.
These define the slope (gradient) of L at coordinates $(w_0, \ldots, w_n)$

Change $w_0, \ldots, w_n$ so that this point moves "downhill"
The **gradient descent** update of one weight:

$$w_i = w_i - \alpha \frac{\partial L}{\partial w_i} = w_i + \alpha \sum_j^N (y - \hat{y}) x_{i,j}$$

α is the learning rate (or step size)
Keep updating until L no longer decreases.

Gradient Descent

$$w_i = w_i + \alpha \sum_{j}^{N} (y - \hat{y}) x_{i,j}$$

Some issues:

- For each update, $y - \hat{y}$ needs to be recalculated for N data points
- Values of $\mathbf{x}$ may vary by dimension (this is bad)
- Sum of errors over data points may be large, causing instability
- Choice of α can be tricky

Very small α can help, but this makes learning slow.
Instead, try to work around these issues by scaling data and using stochastic gradient descent. This allows learning with larger α.

Scaling

Let's say temperature varies from 19°C to 25°C and CO_2 level from 400 ppm to 2000 ppm.

Is CO_2 level 200 times more important than temperature? Are they even comparable?

Fix that by scaling everything to $[-1, 1]$ or $[0, 1]$.

Scaling

Before

x_1	x_2	y
21.86	36.24	0.0059
20.00	18.89	0.0027
22.20	26.82	0.0044
19.23	30.63	0.0042
...	...	...

After

x_1	x_2	y
0.95	0.94	0.91
0.87	0.49	0.42
0.97	0.70	0.69
0.84	0.79	0.66
...	...	...

Stochastic Gradient Descent

Calculate loss over single, randomly picked sample, update weights. This is **stochastic gradient descent** (SGD).

initialize $\mathbf{w}$ with random numbers $[0, 1)$
loop
  pick random sample $x_1, \ldots, x_n, y$ from training set
  calculate $\hat{y}$
  update $w_0 = w_0 + \alpha(y - \hat{y})$
  for i in 1 .. n
    update $w_i = w_i + \alpha(y - \hat{y})x_i$

Repeat for a fixed number of iterations
  or check periodically if L over training set is decreasing.
Variation: select small number of samples at a time (minibatch)

Regularization

Recall Occam's Razor: simplest hypothesis should be preferred

Can add regularization term to loss function

$$L_r = L + \lambda \sum_{i}^{n} w_i^2$$

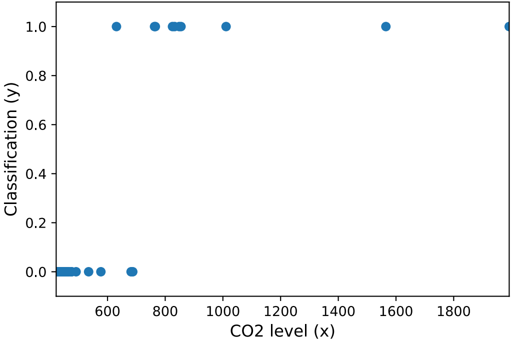
$\lambda > 0$ is regularization parameter.

The general idea is to "shrink" weights
large weights $\iff$ overfitting, divergence

Also, feature selection, as unimportant weights approach 0

Classification Problem

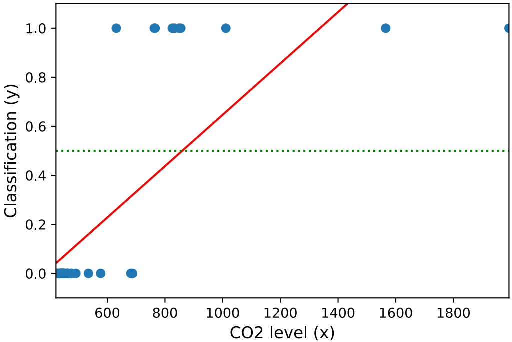
What if the output is binary (0 and 1)?



How to predict y?

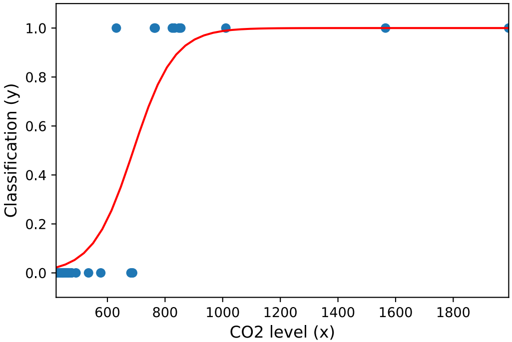
Classification Problem

Straight linear regression sort of works



$$\hat{y} = \begin{cases} 1, & w_0 + w_1 x > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

Logistic Regression



A better fit: the logistic function

$$\hat{y} = \frac{1}{1 + e^{-w_0 - w_1 x}}$$

Logistic Regression

Cannot just plug in w_0, w_1 learned with linear regression,
need to fit the logistic function to the data.

Same idea as before:
update $w_i = w_i - \alpha \frac{\partial L}{\partial w_i}$, so that L decreases.

Logistic Regression

Let logistic function $g(z) = \frac{1}{1+e^{-z}}$
Derivative: $g'(z) = g(z)(1 - g(z))$

Loss gradient, single data point $x_1, \ldots, x_n, y$:

$$\begin{aligned}\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}(y - \hat{y})^2 &= 2(y - \hat{y})\frac{\partial}{\partial w_i}(y - g(\mathbf{w} \cdot \mathbf{x})) \\ &= -2(y - \hat{y})g'(\mathbf{w} \cdot \mathbf{x})x_i\end{aligned}$$

Update rule (ignore constant 2, substitute g'):

$$w_i = w_i + \alpha(y - \hat{y})(1 - \hat{y})\hat{y}x_i$$

(Use this with SGD)

Classification: Probability

Why use logistic regression? Can be shown that

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}\cdot\mathbf{x}}}$$

when:

- x-s are independent (feature selection is important)
- x-s have normal distribution
- y is a binary variable, 1 with probability p and 0 with probability $1 - p$

Classification: Convergence

Linear regression does not converge smoothly with binary y, large errors remain.